A tour of the repl's power mode

Alternate title: "We're doing it live!"

Good News

```
$home/bin/scala -nocompdaemon -e 'println(util.Properties.versionString)'

version 2.7.7.final
real 0m1.841s

version 2.8.1.final
real 0m4.662s

version 2.9.0.1
real 0m4.872s

version 2.10.0.r25052
real 0m2.406s
```

:power

Or: scala -Dscala.repl.power

- Helpful to know some compiler internals
- Only a little bit
- Practically nothing

typer.typedType(tpt).tpe

Actual compiler code.

Inside the Machine

- Trees
- Symbols
- Types
- Phases, Scopes, Contexts...

Opportunities for Confusion

- Printing or otherwise interacting with types can change behavior by forcing lazy types
- The active phase impacts many things
- ...ok there are a few more, but those two can be especially puzzling

```
% scalac -Xshow-phases

    phase name  id  description
    ----------  --  -----------
        parser   1  parse source into ASTs, perform simple desugaring
         namer   2  resolve names, attach symbols to named trees
packageobjects   3  load package objects
         typer   4  the meat and potatoes: type the trees
superaccessors   5  add super accessors in traits and nested classes
       pickler   6  serialize symbol tables
     refchecks   7  reference/override checking, translate nested objects
  selectiveanf   8
      liftcode   9  reify trees
  selectivecps  10
       uncurry  11  uncurry, translate function values to anonymous classes
     tailcalls  12  replace tail calls by jumps
    specialize  13  @specialized-driven class and method specialization
 explicitouter  14  this refs to outer pointers, translate patterns
       erasure  15  erase types, add interfaces for traits
      lazyvals  16  allocate bitmaps, translate lazy vals into lazified defs
    lambdalift  17  move nested functions to top level
  constructors  18  move field definitions into constructors
       flatten  19  eliminate inner classes
         mixin  20  mixin composition
       cleanup  21  platform-specific cleanups, generate reflective calls
         icode  22  generate portable intermediate code
       inliner  23  optimization: do inlining
      closelim  24  optimization: eliminate uncalled closures
           dce  25  optimization: eliminate dead code
           jvm  26  generate JVM bytecode
      terminal  27  The last phase in the compiler chain
```

```
scala> val x = ?[List[_]]
x: power.InternalInfo[List[_]] =
scala.collection.immutable.List

scala> x.
?               allMembers      asInstanceOf    bts
btsmap          companion       declares        erasure
glb             inPackage       info            inherits
isInstanceOf    lub             members         methods
module          overrides       owner           owners
pkg             pkgName         pkgmates        pkgslurp
shortClass      symDef          symName         symbol
toString        tpe             types           whoHas
```

```
scala> class Bippy { implicit def xtoy(x: Int) = x.toFloat }
defined class Bippy

scala> val b = new Bippy
b: Bippy = Bippy@711f6413

scala> import b._
import b._

scala> :implicits
/* 1 implicit members imported from Bippy */
  /* 1 defined in Bippy */
  implicit def xtoy(x: Int): Float

scala> :imports
 1) import java.lang._             (152 types, 157 terms)
 2) import scala._                 (801 types, 809 terms)
 3) import scala.Predef._          (16 types, 167 terms, 96 are implicit)
 4) import power.rutil._           (29 terms)
 5) import b._                     (25 terms, 1 are implicit)
```

```
scala> :implicits
No implicits have been imported other than those in Predef.

scala> import scala.sys.process._
import scala.sys.process._

scala> :implicits
/* 7 implicit members imported from scala.sys.process */
  /* 7 inherited from scala.sys.process.ProcessImplicits */
  implicit def builderToProcess(builder: scala.sys.process.processInternal.JProcessBuilder):
scala.sys.process.ProcessBuilder
  implicit def stringSeqToProcess(command: Seq[String]): scala.sys.process.ProcessBuilder
  implicit def stringToProcess(command: String): scala.sys.process.ProcessBuilder
  implicit def xmlToProcess(command: scala.xml.Elem): scala.sys.process.ProcessBuilder

  implicit def buildersToProcess[T](builders: Seq[T])(implicit convert: (T) =>
scala.sys.process.ProcessBuilder.Source): Seq[scala.sys.process.ProcessBuilder.Source]
  implicit def fileToProcess(file: scala.sys.process.processInternal.File):
scala.sys.process.ProcessBuilder.FileBuilder
  implicit def urlToProcess(url: scala.sys.process.processInternal.URL):
scala.sys.process.ProcessBuilder.URLBuilder
```

```
scala> def timed[T](body: => T): T = {
     |   val start = System.nanoTime
     |   try body
     |   finally println((System.nanoTime - start) + " nanos elapsed.")
     | }
timed: [T](body: => T)T

scala> :wrap timed
Set wrapper to 'timed'

scala> (1 to 1000000).sum
39163000 nanos elapsed.
res0: Int = 1784293664

scala> (1 to 10000000).sum
252608000 nanos elapsed.
res1: Int = -2004260032

scala> (1 to 100000000).sum
1733811000 nanos elapsed.
res2: Int = 987459712

scala> 1733811000 / 39163000
425000 nanos elapsed.
res3: Int = 44
```

REPL